Unsupervised Monocular Depth Estimation Using Atrous Convolutions

Fabian Kessler, Steven Lang, and Dominik Straub

Technische Universität Darmstadt
{firstname.lastname}@stud.tu-darmstadt.de

Abstract

Monocular depth estimation is concerned with computing a dense depth map from a single image but faces difficulties especially at object boundaries. Atrous convolutions have been successfully employed to this end in the task of semantic segmentation. In this paper, we investigate, whether it is also possible to apply atrous convolutions in unsupervised monocular depth estimation. Specifically, we place an Atrous Spatial Pyramid Pooling block in a convolutional neural network between the encoder and decoder. This block allows for computing feature maps at different spatial scales on top of the encoder output. Our experiments show that atrous convolutions in the proposed setup do not improve depth estimation performance. Furthermore, the necessity of a lower output stride after the encoder, such that an increased receptive field size is even applicable, harms runtime and increases memory consumption. Finally, we show that it is possible to reduce the number of channels after the encoder, which reduces the parameter count without impairing predictions.

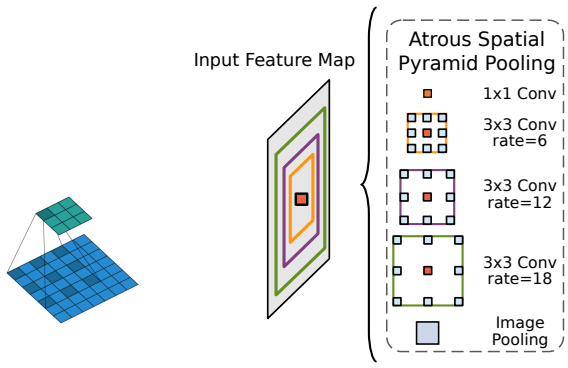
1. Introduction

Depth estimation is one of the most researched areas in computer vision. While traditionally sparse stereo correspondence algorithms were used, most modern work is concerned with estimating dense depth maps from images. Typically, this is achieved via a stereo matching approach, in which (under certain assumptions about epipolar geometry and the appearance of surfaces) a disparity map can be computed by warping one image of a stereo pair into the other. As described by Scharstein & Szeliski [29], this usually involves an appearance cost function, a global smoothness constraint and iterative refinements of depth maps. Most contemporary approaches still make use of these basic ingredients.

Stereo-based approaches have the inherent disadvantages that they require a stereo image pair for each newly generated depth estimate. The respective stereo data, however, is not available for the majority of use-cases. This has sparked research into monocular depth estimation methods, which are usually based on machine learning models trained on large amounts of ground truth depth data. Ground truth depth data, however, is very tedious to capture. It requires the use of expensive and finely calibrated depth scanning hardware such as LiDAR sensors. These introduce further problems including noise in distant samples, sparse measuring, as well as sensor calibration difficulties. Therefore, ground truth depth data is not available in large quantities for most scenarios. The KITTI Stereo dataset from 2015 for example[23], consisting of only 200 images, was captured using a LiDAR sensor and consequently, depth is only available for about 5% of the pixels and only for non-moving objects.

Because of this, recent monocular depth estimation methods [11, 15, 14] approach the estimation process as an unsupervised image reconstruction problem, in which stereo or video data is only required during training and can easily be captured with modern smartphones, for example. The monocular depth estimation architecture by Godard *et al*. [15] demonstrated that when enforcing consistency between two predicted depth maps and the two camera views, superior performance to fully-supervised baselines and generalization to yet unseen datasets can be achieved. Most of their depth estimates' uncertainty can be found in uniform regions and at object boundaries.

This is not surprising, since dense predictions tasks, such as semantic segmentation and depth estimation, face two key challenges: Capturing rich contextual information and detecting objects at multiple scales, while also maintaining sharp object boundaries in the final dense prediction. State of the art CNN architectures typically use small filter sizes, repeated striding operations and max-pooling to down-sample the image, which reduce the resolution of feature maps and as a result make details difficult to recover. As mentioned by Fu *et al*. [9], previous solutions include the



(a) (b)

Figure 1: (a) Atrous convolution with a dilation rate of 2. (b) Atrous Spatial Pyramid Pooling (ASPP). The field of views of the different atrous convolutions are shown in different colors on the input feature map. Figure adapted from Chen *et al*. [3].

use of skip-connections [27], iterative refinement or multi-layer deconvolution networks [24].

In this work, we investigate whether atrous convolutions, which have proven successful in semantic segmentation [3, 4], can also improve monocular depth estimation. Atrous convolutions allow to explicitly control the receptive field size of a filter without introducing additional parameters. This is achieved by skipping pixels at a certain rate (also called filter dilation), and thus inserting zero holes (french: *trous*) into the filter (see Figure 1a). This helps with aggregating information from different spatial scales, while also allowing for larger feature maps, as less downsampling is required in order to achieve a large field of view (FOV).

With these considerations in mind, we designed an architecture using atrous convolutions based on the original monocular depth estimation architecture from Godard *et al*. [15], and we compare this both visually and quantitatively to the original baseline. In the architecture design, we considered predictive performance and runtime and memory consumption. We performed several experiments that show how these factors are influenced by different architectural decisions. We conclude that atrous convolutions, at least in the setups we investigated, do not improve monocular depth estimation. Based on our experiments we propose an architecture that improves upon the baseline performance from Gordard *et al*. [15] while cutting the number of parameters by 24.5%.

2. Related Work

Depth estimation can either be performed on stereo or monocular images, the latter being a more challenging problem. Stereo-based approaches have a long history in computer vision, ranging from more classical methods [29] to recent advancements in deep learning [37]. In this work, we are concerned with monocular depth estimation, where only a single input image is available at test time.

2.1. Monocular Depth Estimation

We give an overview of the two main approaches for monocular depth estimation: supervised methods, which use ground truth depth data, and unsupervised methods, typically using image pairs during training.

2.1.1 Supervised Approaches

Early approaches of monocular depth estimation were based on segmenting images into superpixel patches and then inferring the 3D position and orientation of these patches based on geometric and image-formation assumptions [28, 21]. Most of these methods relied on the use of hand-crafted feature representations such as SIFT [22].

Modern Deep Convolutional Neural Networks (CNN) allow representing the data using richer features that are learned within the network for a specific task. Eigen *et al*. [7, 8] was the first to propose monocular depth estimation as a CNN-based pixel-wise continuous regression problem in order to obtain depth. This work also highlights the importance of multi-scale features, by employing two networks to learn both the global scene structure, along with a second network to refine the depth prediction based on more local features (e.g. walls and objects).

Since then several ideas have been proposed to improve upon the results of a pixel-based regression approach, including the use of multi-scale continuous conditional random fields [33], semantic information [36], attention [18, 5] or edge-specific loss-terms [17]. Jiao *et al*. [18] employ semantic information, in addition to an attention-based loss term that gives higher weight to more distant regions. This setup improves the accuracy of distant depth estimates, as depth estimation networks can suffer from being "short-sighted", due to closer distances being over-represented in the loss. The quality of depth predictions at object boundaries can be improved by fusing information from different spatial scales (multi-scale feature fusion) and employing edge-specific loss-functions using both differences in gradients and angle of surface normals between the estimated depth maps and the corresponding ground truths data [17]. Alternatively, monocular depth estimation can also be formulated as pixel-wise classification [1] or ordinal regression [9]. In this approach, the ranges of possible depths are learned instead of exact depth values. These depth ranges are easier to estimate and allow to reason about the confidence of the predictions and can even lead to superior performance.

2.1.2 Unsupervised Approaches

Unsupervised methods provide an alternative to the supervised methods, in so far that they do not need labeled ground truth depth data, which is not widely available for many application domains and datasets. Instead, these methods rely on estimating disparity maps from either rectified stereo pairs [11, 15] or sequences of videos [30, 14] by learning a mapping between images that correspond to depth or disparity. Importantly, the image pairs are only available during training, which makes the methods monocular at test time. Garg *et al*. [11] suggested directly learning a disparity map through an encoder-decoder CNN, by applying a photo-metric difference loss to compare the reconstructed image, to the true target image. This approach was further refined by Godard *et al*. [15] through learning two disparity maps instead of one and enforcing consistency between both maps, along with the reconstructed images and their respective input images.

More recent work has focused on the use of video data to solve depth estimation by performing view-synthesis based on successive frames [30, 2, 14]. Typically, this involves training a separate pose network to learn the corresponding camera transformations and depth information. Recent work by Godard *et al*. [14] shows that depth features can also help in pose estimation, by sharing weights between the two networks. One drawback of video-based approaches is that they can not effectively deal with moving objects between frames. Objects moving together with the camera are typically projected into infinite depth, such that the photometric loss is lowered. A recent approach [2] has shown that it is possible to explicitly model and account for individual object motion by obtaining both their speed and direction and warping them independent of background motion. This can lead to performance comparable to stereo-based approaches. Gordard *et al*. [14] showed, that the network can be trained with both monocular, as well as stereo data to overcome this problem and achieve state-of-the-art performance.

Furthermore, semi-supervised approaches have been suggested in which both the ground truth depth data, along with the image data is used to ensure image-consistency of the depth maps and correspondence to the ground truth depth [20].

2.2. Atrous Convolutions

Atrous convolutions were originally introduced in the context of wavelet transforms [16] as a means of reducing computational complexity. Since atrous convolutions skip certain pixels, they achieve a larger filter without the need for additional parameters. Recently, atrous convolutions have been heavily used in semantic segmentation [3, 4, 31, 34], with some applications to other tasks such as image classification [35]. When depth estimation is treated as a classification task, recent work suggests that atrous convolutions can also be successfully employed to obtain feature maps at different spatial scales without repeated downsampling of the image [9, 5]. Inspired by these successful applications, we apply atrous convolutions to monocular depth estimation as a regression problem.

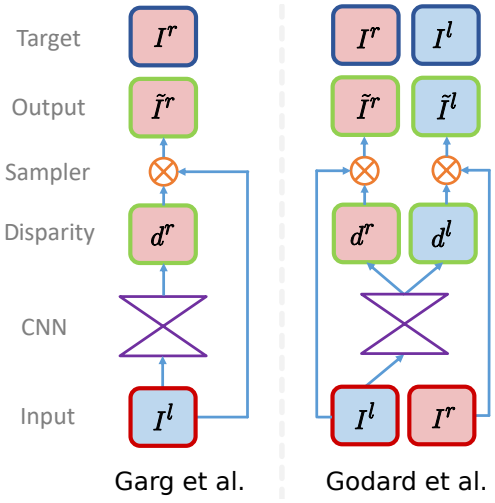


Figure 2: Monocular depth estimation via stereo reconstruction: naive approach by Garg et al. [11] and method with left-right consistency (figure adapted from Godard et al. [15]).

3. Methods

In this section, we briefly describe the approach by Godard *et al*. [15]. We then explain atrous convolutions and how we extended Godard *et al*.'s architecture with atrous convolutions.

3.1. Monocular Depth Estimation

Monocular depth estimation can be treated as a stereo reconstruction problem, where a left an right image (I^l and I^r) are available during training. The left image is fed through an encoder-decoder CNN, which outputs a disparity map w.r.t. the right image (d^r). This disparity map is applied to the left image using backward warping with bilinear interpolation in order to obtain the reconstructed right image $\tilde{I}^r$ (see Figure 2, left column). The CNN can be trained with an appearance-based loss function that measures the photometric difference between I^r and $\tilde{I}^r$.

Godard *et al*.'s [15] main contribution is the idea of left-right consistency: instead generating only the right disparity map, the CNN outputs a pair of corresponding left and right disparity maps (d^l and d^r). The left image is further warped with the right disparity map and vice versa (see Figure 2, right column). The loss function consists of three weighted

terms:

$$C_s = \alpha_{ap} C_{ap} + \alpha_{ds} C_{ds} + \alpha_{lr} C_{lr} \quad . \tag{1}$$

C_{ap} is the appearance-based loss, C_{ds} is a disparity smoothness term, which penalizes high derivatives in the disparity maps, and C_{lr} is a left-right consistency penalty, which ensures that the left and right disparity maps agree. Each term is weighted with an α value respectively. Additionally, this loss is computed and summed across four different output scales: $C = \sum_{s=1}^{4} C_s$.

We use these basic ideas (image reconstruction, smoothness, left-right consistency), but instead of a standard ResNet50 backbone, we experiment with atrous convolutions.

3.2. Atrous Convolutions

The output $g(i)$ at index i of a 1D atrous convolution w is given by

$$g(i) = \sum_{k=1}^{K} x(i + rk) w(k) \quad , \tag{2}$$

where x is the input and K is the filter size. The filter dilation r specifies the rate at which the input is sampled. A standard convolution is a special case of an atrous convolution, where $r = 1$. The notion of an atrous convolution can be generalized to 2D for vision problems straightforward. Figure 1a shows a 2D atrous convolution with $r = 2$.

We employ the idea of an Atrous Spatial Pyramid Pooling (ASPP) block [3], which contains several atrous convolutions with different atrous rates in parallel (see Figure 1b). This is motivated by classical image pyramid methods [32, 26], which process images at different spatial scales. Since varying atrous rates results in filters of different spatial dimensions, an ASPP block resembles a feature map at different spatial scales. The atrous convolutions, along with a global averaging pooling layer, are concatenated and passed through a 1×1 convolution, which reduces the number of channels to 256.

3.3. Output Stride

The output stride of a feature map is the factor by which its spatial dimensions are smaller than those of the input image. Each ResNet block changes the output stride by a factor determined by the stride of its main convolutional layer. In a standard ResNet50, each ResNet block has a stride of 2, which yields an output stride of 64 after the encoder. This was also used in the monocular depth estimation architecture by Godard *et al.*, resulting in a 4×8 feature map after the encoder for 256×512 input images. But in order to obtain feature responses at different spatial scales using atrous convolutions (with varying kernel size and thus FOVs), a certain size of image is required as illustrated in Figure 3.

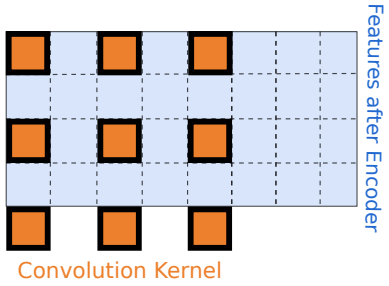


Figure 3: A 4×8 feature map processed with an atrous convolution with rate 2. Some of the filter weights are out of the image range, which results in boundary issues. This effect is even stronger for convolutions with larger atrous rates.

Hence, we made the output stride in our network architecture variable by adapting the ResNet block output strieds.

3.4. Our Architecture

Figure 4 shows a sketch of our architecture. We use the same basic setup as Godard *et al.* [15], i.e. a ResNet50 backbone, skip connections between the encoder and the decoder and output disparities at multiple spatial scales. Our main contribution is the extension of the encoder by an ASPP block, inspired by the DeepLab architecture [3]. For our main experiments, we insert the ASPP module after the final layer of the ResNet50, since this placement has proven to work well in other work [4, 9]. Alternative choices, e.g. in between ResNet blocks, are also possible (see Section 4.5 and supplement A).

The ASPP block introduces additional parameters to the architecture. By processing the ASPP block output with a 1×1 convolution, we reduce the number of feature maps to 256 after the encoder, compared to 2048 feature maps in the original architecture. Since this reduces the decoder input dimensionality, we effectively keep the number of parameters identical to the original architecture at 58.4 million[1].

4. Results

The encoder-decoder network was implemented using PyTorch [25] (version `1.0.1`) and training took place on a single GPU (NVIDIA GeForce GTX 1080 Ti or GTX TITAN X). For all our experiments we used the KITTI data set [12] with a total number of 30159 images from which 29000 were used during training and 1159 for validation. Additionally, the KITTI Stereo 2015 dataset [23] provided 200 ground truth disparities which were used for numerical comparisons between models. This evaluation was performed by converting both, the predicted disparities, as well

[1]While inspecting the TensorFlow implementation by Godard *et al.*, we found that their ResNet50-based architecture, in fact, had 58.4 million parameters, instead of the specified 48 million.

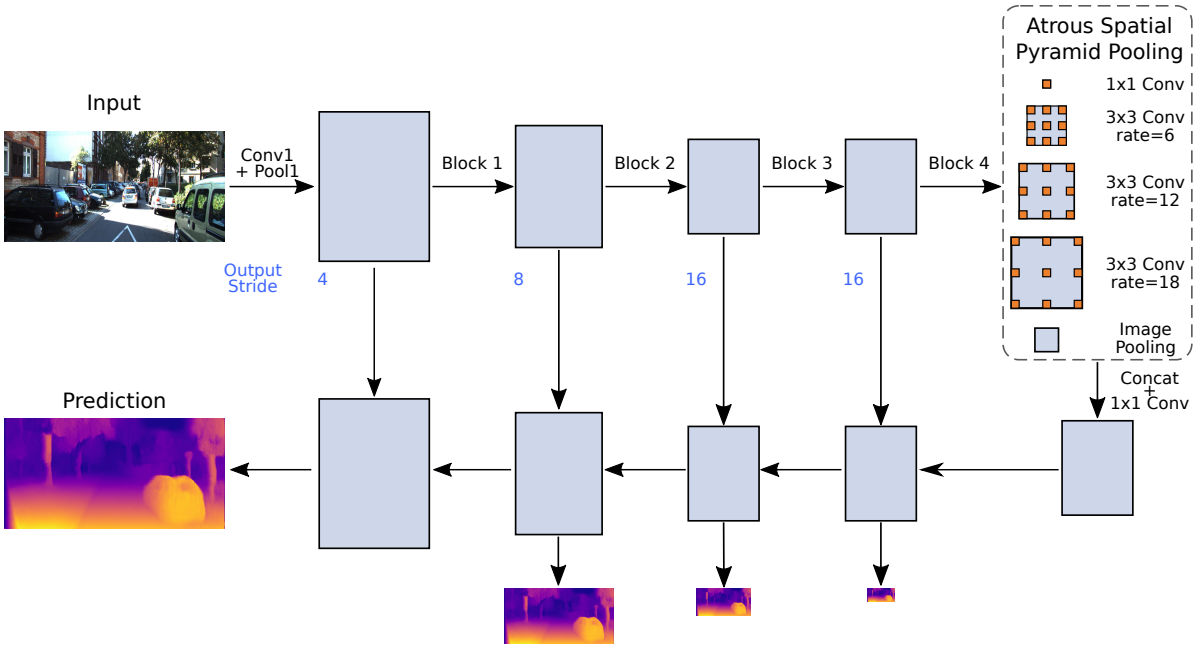


Figure 4: Overview of our architecture. The input image is fed through a ResNet50 architecture, consisting of four blocks. Final feature maps of ResNet50 are passed to the ASPP block, which processes its input at several spatial scales in parallel. The decoder then upsamples the image again, while taking low-level features from the skip-connections and outputting disparity maps at multiple spatial scales.

as the sparse ground truth disparities into depth space and capping depth to 80 meters.

We used metrics for numerical evaluation found in Eigen *et al*. [8] and present the absolute relative error for our experiments. A table with all metrics for all experiments can be found in Supplement D. Additionally, to bring the results of the following sections into perspective, we repeated the baseline implementation 12 times with different random seeds and measured the mean and standard deviation of the error metrics. For the abs. rel. error e.g., we observed a mean of 0.1125 and a standard deviation of 8.581×10^{-4}. We also noticed about 25% of the runs were not able to learn, with the loss staying constant at the initial value. Thus the networks performance was not robust to weight initialization.

Most of the training parameters were adopted from Godard *et al*., including the number of epochs, learning-rate, optimization procedure, and data augmentation. Specifically we trained our network for 50 epochs using the Adam optimizer [19] ($\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$). The initial learning rate of $\lambda = 10^{-4}$ was adjusted according to the batch size (i.e. doubling the learning rate when doubling the batch size). Separate experiments confirmed that batch size did not affect training results significantly when also doubling the learning rate, hence we decided to use a batch size of 16 and thus $\lambda = 2 * 10^{-4}$ for our initial learning rate. This learning rate stayed constant for the first 30 epochs of training and was then halved every 10 epochs for the rest of the training. The training duration was 20 hours on average and mostly depended on the GPU and the output stride. Values for the loss weight hyperparameters α from Section 3.1 were inherited from the original paper. For these, no further hyperparameter optimizations have been conducted.

For training, the KITTI images were resized from 1275×375 to 512×256 and the same data augmentation as in the original paper was performed. This included flipping both images horizontally at 50% chance level, as well as uniformly distributed gamma [0.8,1.2], brightness [0.5,2.0] and color shifts [0.8,1.2].

For our final selection of models, we also employed cityscapes [6] pretraining, followed by fine-tuning on KITTI. Here we used the same learning-rate and epoch count for both pretraining and fine-tuning.

4.1. Baseline: Implementation Issues

Before running experiments with architectural changes, we replicated the original publication's [15] results. During this process, we encountered some difficulties: Using the standard ResNet50 architecture from the PyTorch package as the encoder resulted in vastly underestimated depth in the uniform sky regions of the KITTI dataset (see Figure 5, middle row). This was not present in the disparity

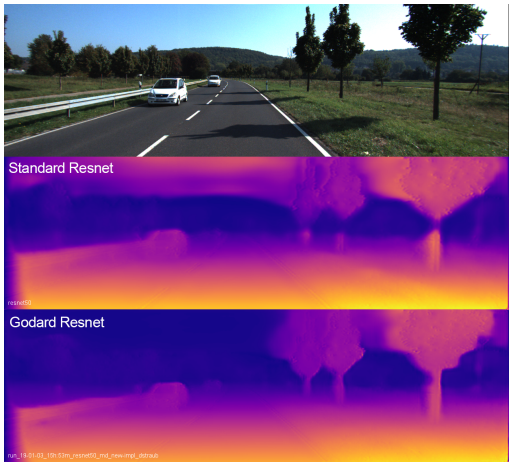


Figure 5: Comparison between a standard ResNet50 and Godard *et al.*'s ResNet50. Disparities from a standard ResNet50 show severe artifacts in the uniform sky regions, while the modified ResNet50 does not exhibit these problems.

Stride	Abs. Rel.	Runtime ($^{\text{min}}/_{\text{epoch}}$)	Memory (GB)	Param. (M)
64	0.1120	**15**	**6.12**	58.4
32	0.1048	29	8.87	58.4
16	**0.1041**	29	8.88	58.4
8	0.1068	43	10.61	58.4

Table 1: Baseline model [15] with different output strides (64 being the default): the absolute relative error is optimal at an output stride of 16, while the runtime and memory consumption monotonically increase when decreasing the output stride.

maps provided by the original authors [15], which prompted us to investigate Godard *et al.*'s ResNet50 implementation more thoroughly. We found the following three differences between their implementation compared to a standard ResNet50: 1) Lack of batch normalization, 2) using nearest neighbor instead of linear upsampling in the decoder and 3) switching the order of convolutions with $stride = 1$ and $stride = 2$ inside a ResNet block. While these differences were tested in isolation, we could not figure out which caused the different behavior of the networks. However, porting the original TensorFlow implementation line by line to PyTorch fixed the erroneous estimations in homogeneous regions, such as the sky (see Figure 5, bottom row). Therefore, we used the modified ResNet50 implementation in all following experiments.

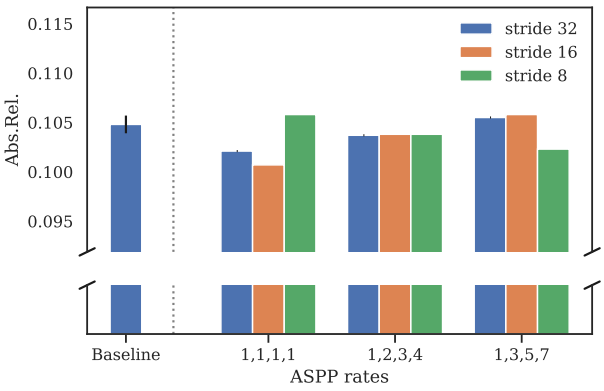


Figure 6: Absolute relative error of architectures with different ASPP rates. A rate setup of [1,2,3,4] stands for four ASPP modules with an ASPP rate of 1, 2, 3 and 4 respectively.

4.2. Output Stride

Since atrous convolutions can only be effective for images of a certain resolution, which allows the whole kernel to be inside the image, we needed to decrease the output stride of the network (i.e. increase the input size to the ASPP). We conducted an experiment with varying output strides in the baseline model in order to isolate the output stride effect, such that the effect of the ASPP is really due to the ASPP and not merely due to the decreased output stride. We tested four different output strides (namely 8, 16, 32 and 64) for their effect on predictions and computation time required per epoch. The results in Table 1 show that increasing the output stride up to 16 decreases the abs. rel. error on the evaluation, while increasing the training time required for one epoch. The abs. rel. error increased for output stride 8, among a substantially longer training time. Unless otherwise specified, we focused on output stride 32 for the following experiments, as this provided the best trade-off between prediction performance and computation time.

4.3. ASPP Rates

In this set of experiments we tested the impact of atrous convolutions with varying atrous rates on the predictive performance of the network. We ran a total of three configurations with atrous rates up to 7 on networks with both output stride 16 and 32 and compared this to the baseline model described in Section 3.1. Figure 6 shows the abs. rel. error for the different configurations.

The results suggest that the use of atrous convolutions in the described setup harms the depth estimation performance as seen by the increase in the error. Visual comparison of reconstruction error maps (see supplementary B.1) also show

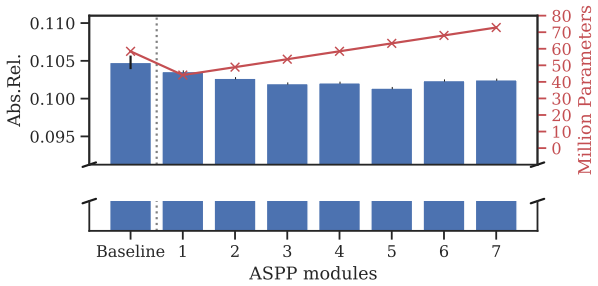


Figure 7: Absolute relative error of architectures with a different number of ASPP modules. The error bar indicates the standard deviation of the abs. rel. error of our baseline implementation. Additionally, the number of network parameters is plotted for each architecture.

no apparent differences between the different atrous rates. The model which uses four standard convolutions (*rate = 1*) performed best against models with an increasing atrous rate, and is even slightly superior to the baseline. For an output stride of 8, the effect's direction is inverted, which suggests that atrous convolutions might help on larger images. This was not investigated further, since models with output stride 8 took significantly longer to train and both output stride 16 and 32 outperform the output stride 8 model.

4.4. Number of ASPP Modules

Since the network with four standard convolutions in parallel inside the ASPP performed best, we investigated the impact of the number of these convolutional layers. Figure 7 shows the abs. rel. error for models with increasing number of standard convolutions in the ASPP, along with the number of parameters. One can see, that adding one convolutional layer increases the parameter count by about 4 million, while slightly improving error metrics on the evaluation. This trend starts to decline after more than 5 convolutions are added.

4.5. Atrous Convolutions in the Encoder

In the previous, we saw that adding an ASPP module at the end of the encoder, which was successfully done in the DeepLab Architecture for segmantic segmentation [3, 4], impaired depth estimation when atrous rates larger than 1 were used. Therefore, we asked whether depth prediction would benefit from employing atrous convolutions with different rates between or within earlier layers of the encoder (see supplementary Figure A.1 and Figure A.2).

In the first experiment we placed the entire ASPP module between ResNet block 1 and 2 to obtain feature responses at different spatial scales at an earlier stage of the encoder. At this stage the image has only been resized to 64×32 corresponding to an output stride of 8. As an initial exploration, we compared a model with only standard convolutions (rates [1,1,1,1]) against a model with atrous rates [1,6,12,18] (the default values in DeepLab [3]). The model with only standard convolutions performed better than the one with atrous convolutions (see supplementary Table D.4). Thus, we did not further explore this direction and focused on placing the ASPP after the encoder.

In the second experiment we employed atrous convolutions within the ResNet blocks themselves. Again, all other ASPP rates larger than 1 harmed the depth prediction performance as can be seen by an increase in the error metrics compared to the baseline (see supplementary Table D.5).

4.6. Evaluation on Virtual KITTI

Our goal is to evaluate whether atrous convolutions help with monocular depth estimation, especially if they improve the sharpness at object boundaries. Since the KITTI Stereo 2015 ground truth data is rather sparse, it might not contain data points at exactly those most relevant positions at object boundaries. Synthetic data, on the other hand, provides exact ground truth depth at every single pixel. For this reason, we evaluated the models presented in the previous sections on the Virtual KITTI (VKITTI) dataset [10]. Because it only provides monocular images, we could not use it for training but only for evaluation. VKITTI contains clones of five sequences from the original KITTI Vision Benchmark Suite [13] in different weather conditions. To keep these weather conditions identical to the training set, we only chose those sequences that are exact clones of the original sequences, yielding 2126 images in total.

Figure 8 shows disparity error maps for the models with different atrous rates from Section 4.3 at output stride 32. As on KITTI, all models have problems predicting disparity accurately at the object boundaries. Upon visual inspection of the disparity maps, there is no apparent difference between the models with different atrous rates. Numerical evaluation did also not provide any clear conclusions w.r.t. the effectiveness of atrous convolutions.

4.7. Improvement over Baseline

Summarizing the previous experiments, Table 2 shows our most interesting models in comparison to the baseline model of Godard *et al*. [15] and illustrates some trade-offs when making architectural decisions. Our best performing model, using an output stride of 16 and four ASPP modules with an atrous rate of 1, achieves an improvement of 4.43% in the abs. rel. error over the baseline model while having the same number of parameters (model A). There is, however, no clear improvement in terms of visual inspection of the disparity maps (see Figure 9. When we further degenerate the ASPP block to only use the very first 1×1 module, we can reduce the number of parameters by 24.5%,

Model	Output-stride	ASPP	Abs.Rel.	Sq.Rel.	RMSE	Log RMSE	a1	a2	a3	Params (M)	Δ Abs. Rel. (%)
Godard	64	-	0.0970	0.8960	5.093	0.176	0.962	0.962	0.986	58.4	–
Godard	16	-	0.0955	0.8752	4.937	0.174	0.881	0.961	0.984	58.4	2.57
Ours (A)	16	1-1-1-1	**0.0927**	**0.8132**	**4.865**	**0.168**	**0.888**	**0.967**	**0.987**	58.4	**4.43**
Ours (B)	16	1	0.0944	0.8446	5.011	0.174	0.884	0.963	0.986	**44.1**	2.68

Table 2: Final model comparison. Each model was pretrained on the Cityscapes dataset and evaluated on KITTI Stereo 2015 after applying postprocessing as in [15]. The last column measures the relative difference of the abs. rel. error to the original model from Godard *et al*. [15].

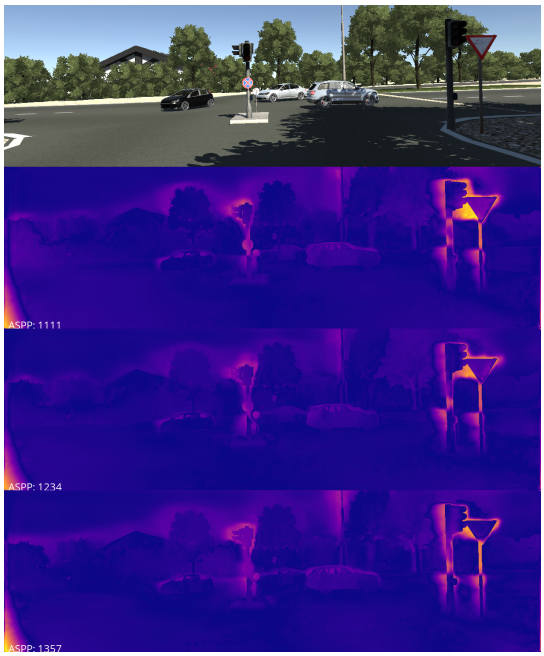


Figure 8: Disparity error maps (absolute difference between predicted and ground truth disparity) on VKITTI. Atrous rates from top to bottom: [1,1,1,1], [1,2,3,4], [1,3,5,7]. Error maps for additional images are available in the supplementary Figure C.2.

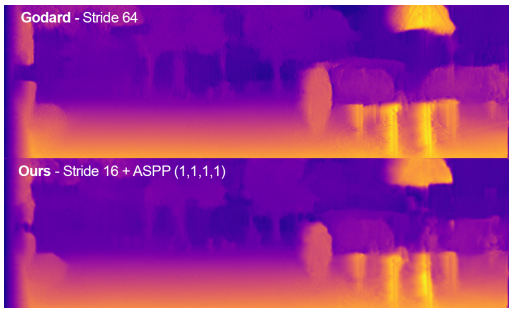


Figure 9: Disparity map comparison of our best model to the baseline [15]. There are no obvious visual differences.

while still achieving an improvement of 2.68% over Godard *et al*. [15] (model B).

5. Conclusion

We have presented the idea and results for the incorporation of atrous convolutions into Godard *et al*.'s [15] unsupervised monocular depth estimation framework using left-right consistency. Experiments with atrous convolutions were mainly conducted using the Atrous Spatial Pyramid Pooling block, inserted between the encoder and decoder, which was already successful in the task of image segmentation [3]. We conclude, that atrous convolutions within the presented experiments do not improve monocular depth estimation. Furthermore, the use of atrous convolutions enforces a higher output stride which consequently harms runtime and memory consumption.

In the event of these experiments we additionally found out, that in Godard *et al*.'s [15] architecture, it is possible to reduce the number of channels between the encoder and decoder. This can decrease the number of network parameters and improve runtime, without losing predictive power.

Our experiments focused on architectural design instead of hyperparameter tuning. Finding the optimal set of hyperparameters for models with atrous convolutions might lead to different results. Future work should investigate why atrous convolutions work so well in semantic segmentation, but cannot be applied straightforwardly to continuous depth estimation. In semantic segmentation, there is a clear distinction between object classes, while regression problems face the harder problem of predicting a continuous signal. Recent work suggests that atrous convolutions help if depth estimation is treated as a classification problem with discrete depths [9]. Alternatively, semantic features could be leveraged to improve depth estimation performance [18], in which case atrous convolutions might be helpful.

References

[1] Y. Cao, Z. Wu, and C. Shen. Estimating Depth from Monocular Images as Classification Using Deep Fully Convolutional Residual Networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3174–3182, 2018. 2

[2] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova. Depth Prediction Without the Sensors: Leveraging Structure for Unsupervised Learning from Monocular Videos. 2018. 3

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018. 2, 3, 4, 7, 8

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018. 2, 3, 4, 7

[5] Y. Chen, H. Zhao, and Z. Hu. Attention-based Context Aggregation Network for Monocular Depth Estimation. pages 1–11, jan 2019. 2, 3

[6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[7] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015. 2

[8] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 2, 5

[9] H. Fu, M. Gong, C. Wang, K. Batmanghelich, H. Fu, M. Gong, C. Wang, K. Batmanghelich, D. Tao, D. Ordinal, H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep Ordinal Regression Network for Monocular Depth Estimation To cite this version : HAL Id : hal-01741163 Deep Ordinal Regression Network for Monocular Depth Estimation. 2018. 1, 2, 3, 4, 8

[10] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016. 7

[11] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016. 1, 3

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 4

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 7

[14] C. Godard, O. Mac Aodha, and G. Brostow. Digging Into Self-Supervised Monocular Depth Estimation. 2018. 1, 3

[15] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 2, 3, 4, 5, 6, 7, 8

[16] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In J.-M. Combes, A. Grossmann, and P. Tchamitchian, editors, *Wavelets*, pages 286–297, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. 3

[17] J. Hu, M. Ozay, Y. Zhang, and T. Okatani. Revisiting Single Image Depth Estimation: Toward Higher Resolution Maps with Accurate Object Boundaries. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1043–1051, 2018. 2

[18] J. Jiao, Y. Cao, Y. Song, and R. Lau. Look deeper into depth: monocular depth estimation with semantic booster and attention-driven loss. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11219 LNCS, pages 55–71, 2018. 2, 8

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[20] Y. Kuznietsov, J. Stückler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. *CoRR*, abs/1702.02706, 2017. 3

[21] Y. H. Lin, W. H. Cheng, H. Miao, T. H. Ku, and Y. H. Hsieh. Single image depth estimation from image descriptors. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 809–812, 2012. 2

[22] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1253–1260. IEEE, jun 2010. 2

[23] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 4

[24] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, 2015. 2

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017. 4

[26] L. H. Quam. Hierarchical warp stereo. In *Readings in computer vision*, pages 80–86. Elsevier, 1987. 4

[27] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015. 2

[28] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2009. 2

[29] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002. 1, 2

[30] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey. Learning Depth from Monocular Videos using Direct Methods. Technical report, 2017. 3

[31] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1451–1460. IEEE, 2018. 3

[32] A. Witkin, D. Terzopoulos, and M. Kass. Signal matching through scale space. *International Journal of Computer Vision*, 1(2):133–144, Jun 1987. 4

[33] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. Monocular depth estimation using multi-scale continuous crfs as sequential deep networks. *CoRR*, abs/1803.00891, 2018. 2

[34] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 3

[35] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3

[36] P. Zama Ramirez, M. Poggi, F. Tosi, S. Mattoccia, and L. Di Stefano. Geometry meets semantic for semi-supervised monocular depth estimation. In *14th Asian Conference on Computer Vision (ACCV)*, 2018. 2

[37] J. Zbontar, Y. LeCun, et al. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016. 2

Unsupervised Monocular Depth Estimation Using Atrous Convolutions
Supplementary Material

A. Atrous Convolutions in the Encoder

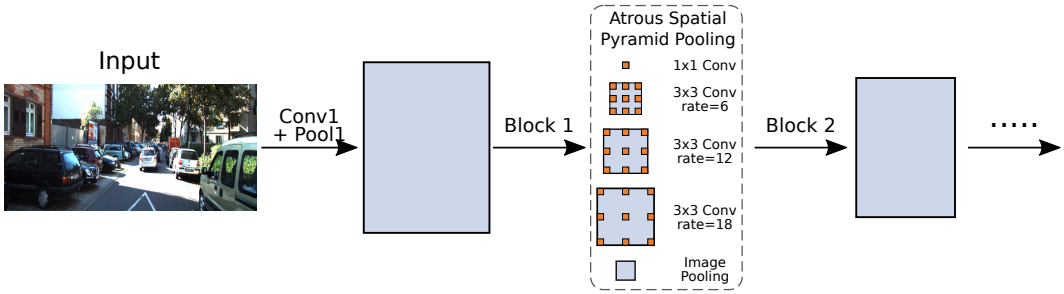


Figure A.1: ASPP after the first ResNet block.

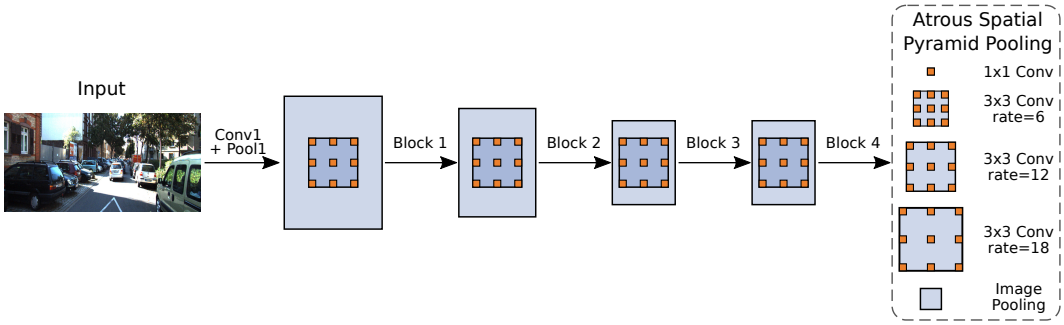


Figure A.2: Atrous convolutions inside the ResNet blocks.

B. Reconstruction error plots

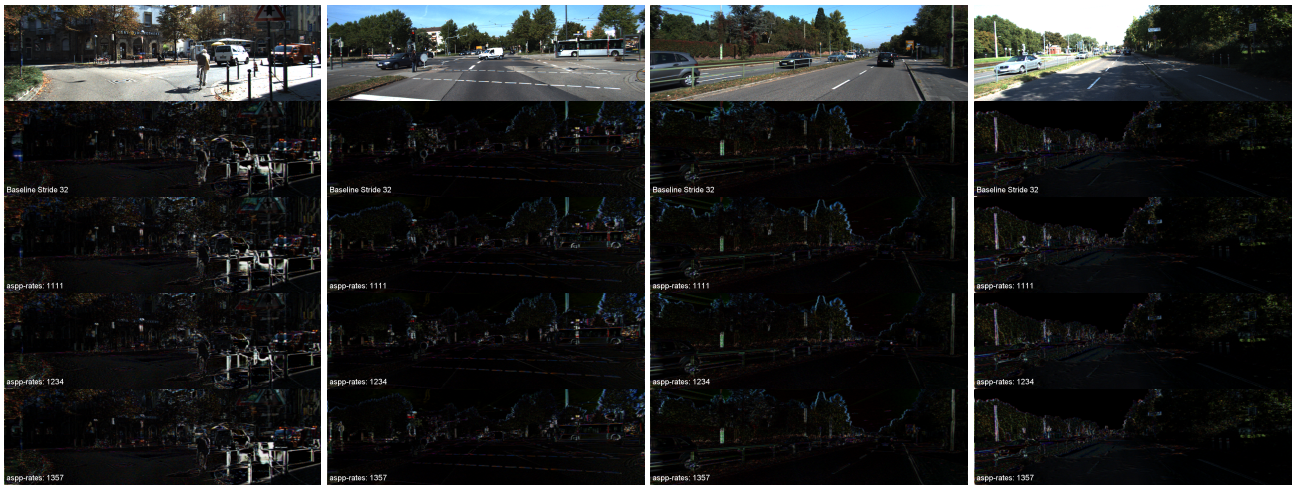


Figure B.1: Reconstruction error for the experiment in Section 4.3 in which the impact of different atrous rates ([1,1,1,1], [1,2,3,4], [1,3,5,7]) was compared to the baseline.

C. VKITTI Error Maps

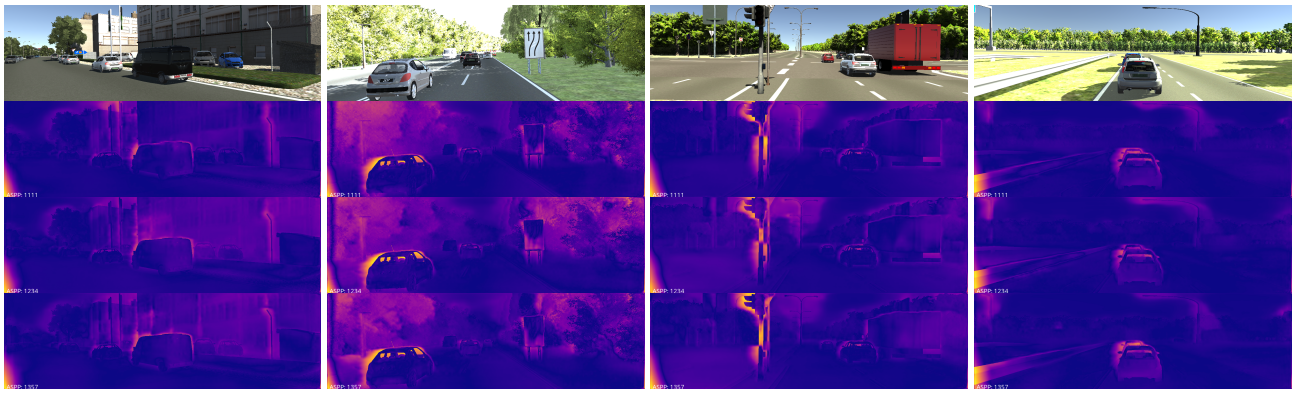


Figure C.2: Disparity error maps on VKITTI (cf. Section 4.6), in which the impact of different atrous rates ([1,1,1,1], [1,2,3,4], [1,3,5,7]) are compared.

D. Full Experimental Results

Output-stride	Abs.Rel.	Sq.Rel.	RMSE	Log RMSE	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params (M)
8	0.1068	1.0408	5.533	0.197	0.859	0.946	0.979	58.4
16	0.1041	1.0097	5.362	0.191	0.865	0.951	0.981	58.4
32	0.1048	1.0353	5.458	0.191	0.866	0.949	0.980	58.4
64	0.1120	1.2025	5.716	0.201	0.854	0.948	0.981	58.4

Table D.1: Comparison of output stride 8, 16, 32 and 64.

Output-stride	ASPP	Abs.Rel.	Sq.Rel.	RMSE	Log RMSE	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params (M)
16	1-1-1-1	0.1007	0.9439	5.322	0.186	0.871	0.956	0.983	58.4
16	1-2-3-4	0.1038	0.9571	5.351	0.187	0.867	0.956	0.980	58.4
16	1-3-5-7	0.1058	0.9803	5.472	0.192	0.864	0.95-	0.981	58.4
32	1-1-1-1	0.1021	0.9561	5.300	0.185	0.870	0.956	0.984	58.4
32	1-2-3-4	0.1037	1.0007	5.446	0.190	0.869	0.954	0.983	58.4
32	1-3-5-7	0.1055	1.0374	5.421	0.191	0.866	0.951	0.982	58.4

Table D.2: Experiment on the influence of increasing atrous rates in the ASPP block for output stride 16 and 32. For both output strides, increasing the atrous rates has increased the depth estimation error on all metrics.

Output-stride	ASPP	Abs.Rel.	Sq.Rel.	RMSE	Log RMSE	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params (M)
32	1	0.1036	1.0123	5.403	0.187	0.867	0.955	0.984	44.1
32	1-1	0.1027	1.0260	5.373	0.188	0.872	0.955	0.983	48.9
32	1-1-1	0.1020	0.9639	5.356	0.188	0.868	0.956	0.983	53.6
32	1-1-1-1	0.1021	0.9561	5.300	0.185	0.870	0.956	0.984	58.4
32	1-1-1-1-1	0.1014	0.9703	5.303	0.186	0.873	0.955	0.984	63.3
32	1-1-1-1-1-1	0.1024	0.9893	5.419	0.189	0.868	0.954	0.982	68.0
32	1-1-1-1-1-1-1	0.1025	0.9820	5.341	0.869	0.957	0.984	0.983	72.8

Table D.3: Number of ASPP Modules. The evaluation results become better the more parallel convolutions are used in the ASPP. The point of diminishing returns is reached after about 7 convolutions. The number of parameters increases by about 4.8 million for each additional convolution.

Output-stride	ASPP	Abs.Rel.	Sq.Rel.	RMSE	Log RMSE	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params (M)
32	1-1-1-1	0.1065	1.0612	5.519	0.194	0.861	0.951	0.982	60.7
32	1-6-12-18	0.1108	1.1424	5.615	0.197	0.858	0.947	0.980	60.7

Table D.4: ASPP after the first ResNet block. The model with only standard (*atrous rate* $= 1$) convolutions performed better.

Output-stride	Atrous rates	Abs.Rel.	Sq.Rel.	RMSE	Log RMSE	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params (M)
16	1-1-1-1	0.1007	0.9439	5.322	0.186	0.871	0.956	0.983	58.4
16	1-2-1-2	0.1028	0.9788	5.360	0.188	0.867	0.954	0.983	58.4
64	1-1-1-2	0.1110	1.0958	5.627	0.197	0.857	0.949	0.981	58.4
64	1-2-1-2	0.1135	1.1240	5.675	0.200	0.851	0.947	0.980	58.4

Table D.5: Atrous convolutions inside the ResNet blocks. This experiment was conducted two different output strides (64 and 16). We changed the atrous rates of the main convolutional layer in the four ResNet blocks. Less atrous convolutions lead to better numerical scores.